1. Alphabet

a. Upper case and lower case letters (a-zA-Z)

b. Decimal digits (0-9)

c. Underline character (_)

2. Lexical

a. Special symbols

- operators: + - * / % = < <= == >= > != && || ! ++ --

- separators: [ ] ( ) {} , : ; space \n \t

- reserved words: int string char bool return while do for if else read write struct or and true false START END

b. Identifiers

identifier ::= letter | letter{letter}{digit} -> ex: x1, a, c2a etc.

letter ::= “a” | “b” | … | “z” | “A” | “B” | … | “Z”

digit ::= “0” | “1” | “2” | … | “9”

nonZeroDigit ::= “1” | “2” | … | “9”

zero ::= “0”

sign ::= [“+” | “-“]

character ::= letter | digit | “.” | “_”

c. Constants

int - rule

int := [sign] nonZeroDigit {digit} | zero

char - rule

char := ‘character’

string - rule

string := “{character}”

bool - rule

bool := true | false

3. Syntax

program ::= “START” statement “END”

```
statement ::= singleStatement | singleStatement statement

singleStatement ::= declarationStatement | assignmentStatement | conditionalStatement |
                    loopStatement | writeStatement | readStatement | structStatement |
                    incrementStatement


declarationStatement ::= primitiveDeclaration | arrayDeclaration “;”

primitiveDeclaration ::= type var

arrayDeclaration ::= type identifier “[“ number “]”

type ::= “int” | “char” | “string” | “bool”

var = identifier | identifier “,” var

number ::= nonZeroDigit {digit}


assignmentStatement ::= identifier “=“ expression “;”

incrementStatement ::= identifier “++” | “--” “;”

conditionalStatement ::= “if” “(“ condition “)” “{“ statement “}” | “if”
          “(“ condition “)” “{“ statement “}” “else” “{“ statement “}”

loopStatement ::= “while” “(“ condition “)” “{“ statement “}”

writeStatement ::= “write” “(“ expression ”)” “;”

readStatement ::= “read” “(“ identifier “)” “;”


condition ::= expression relation expression | condition logicalOperator condition |
                    expression | “!” condition

relation ::= “<“ | “<=“ | “>=” | “>” | “==“ | “!=“

logicalOperator ::= “&&” | “||” | “and” | “or”

expression ::= expression arithmeticOperator term | term

arithmeticOperator ::= “+” | “-“ | “*” | “/“ | “%”

term ::= “(“ expression “)” | identifier | constant

constant ::= int | string | char | bool


structureDeclaration ::= “struct” identifier “{“ declarationList “}”

declarationList ::= declarationStatement | declarationStatement declarationList
```